

# Unidade VI:


## Ordenação Interna - Quicksort



**PUC Minas**

Adaptação dos slides elaborados pelo Instituto de Ciências Exatas e  
Informática - Departamento de Ciência da Computação

- Funcionamento básico
- Algoritmo
- Análise do número de comparações
- Escolha do Pivô
- Conclusão

- **Funcionamento básico** 
- Algoritmo
- Análise do número de comparações
- Escolha do Pivô
- Conclusão

# Introdução

- Proposto por Hoare em 1960 e publicado em 1962
- Algoritmo de ordenação mais rápido para a maioria das situações
- Provavelmente, ele é o mais utilizado
- Algoritmo do tipo dividir para conquistar

# Funcionamento Básico


- Divide o *array* em duas partes que serão independentemente ordenadas e a combinação de seus resultados produz a solução final
  - A parte da esquerda terá elementos menores ou iguais a um pivô
  - A parte da direita terá elementos maiores ou iguais a um pivô

# Funcionamento Básico

- Particionamento:
  - Escolha arbitrariamente um pivô
  - Percorra o *array* a partir da esquerda enquanto  $array[i] < \text{pivô}$
  - Percorra o *array* a partir da direita enquanto  $array[j] > \text{pivô}$
  - Se  $i \leq j$  então troque  $array[i]$  com  $array[j]$
  - Continue o processo enquanto  $i \leq j$

# Funcionamento Básico

- No final do particionamento, o *array* estará particionado de tal forma que:
  - Os elementos *array*[esq], *array*[esq+1], . . . , *array*[j] são  $\leq$  que pivô
  - Os elementos *array*[i], *array*[i+1], . . . , *array*[dir] são  $\geq$  que pivô

- Funcionamento básico
- **Algoritmo** 
- Análise do número de comparações
- Escolha do Pivô
- Conclusão



# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

Considere que Trocar(i, j) são os seguintes comandos:

```

int temp = array[i];
array[i] = array[j];
array[j] = temp;

```

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Algoritmo

```
void Quicksort(int[] array, int esq, int dir) {  
    int i = esq, j = dir, pivo = array[(esq+dir)/2];  
    while (i <= j) {  
        while (array[i] < pivo)  
            i++;  
        while (array[j] > pivo)  
            j--;  
        if (i <= j)  
            { Trocar(i, j); i++; j--; }  
    }  
    if (esq < j)  
        Quicksort(array, esq, j);  
    if (i < dir)  
        Quicksort(array, i, dir);  
}
```

Na primeira chamada, teremos:  
Quicksort(array, 0, n-1)

sendo  $n$ , o número de  
elementos no array

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Algoritmo

```
void Quicksort(int[] array, int esq, int dir) {  
    int i = esq, j = dir, pivo = array[(esq+dir)/2];  
    while (i <= j) {  
        while (array[i] < pivo)  
            i++;  
        while (array[j] > pivo)  
            j--;  
        if (i <= j)  
            { Trocar(i, j); i++; j--; }  
    }  
    if (esq < j)  
        Quicksort(array, esq, j);  
    if (i < dir)  
        Quicksort(array, i, dir);  
}
```

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>															<b>dir</b>

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

i

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>															<b>dir</b>

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
    
```

</															

# Algoritmo

pivô

10

**$(0 + 15) / 2: 7$**

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

<b>0</b>	<b>1</b>	<b>5</b>	<b>3</b>	<b>15</b>	<b>16</b>	<b>9</b>	<b>10</b>	<b>4</b>	<b>3</b>	<b>30</b>	<b>5</b>	<b>20</b>	<b>48</b>	<b>71</b>	<b>82</b>
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>														<b>dir</b>	

# Algoritmo

pivô

10

0 <= 15: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir



# Algoritmo

pivô

10

0 < 10: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

1 < 10: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

5 < 10: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

			i													j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq															dir	

# Algoritmo

pivô

10

3 < 10: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

				i													j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq															dir		



# Algoritmo

pivô

10

15 < 10: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

				i													j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq																	dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

82 > 10: true

				i													j
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq																	dir

# Algoritmo

pivô

10

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);       i++;    j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
    
```

				i											j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

## Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

71 > 10: true

				i											j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

<div><div></div><div>i</div><div>j</div></div>																
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

48 > 10: true

<div><div></div><div>i</div><div>j</div></div>																
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

				i									j				
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq												dir					

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

20 > 10: true

				i												j	
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq														dir			



# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

				i						j					
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

5 > 10: false

i											j						
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		
esq																dir	

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

4 <= 11: true

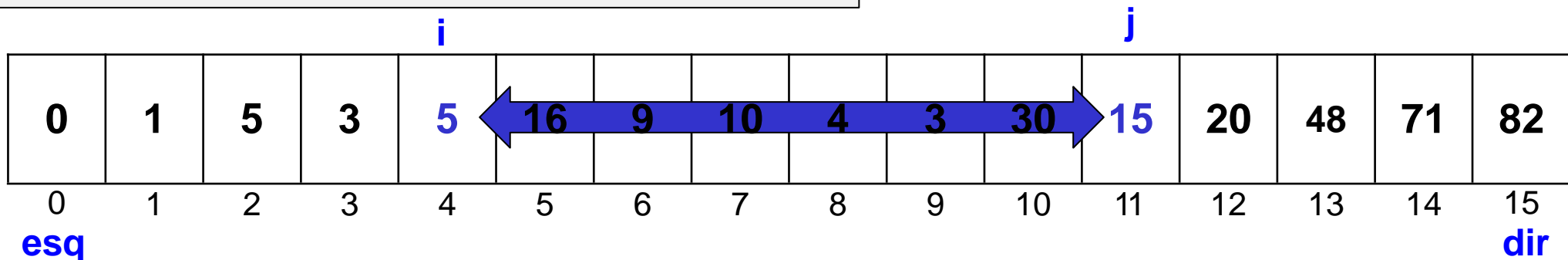
				i					j						
0	1	5	3	15	16	9	10	4	3	30	5	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq												dir			

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```



# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

					<i>i</i>							<i>j</i>				
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
<i>esq</i>																<i>dir</i>

## Algoritmo

pivô

10

5 &lt;= 10: true

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq					i		j		dir						

# Algoritmo

pivô

10

16 < 10: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

					i						j					
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																
															dir	

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

30 > 10: true

					i					j					
0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir



# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

3 > 10: false

0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

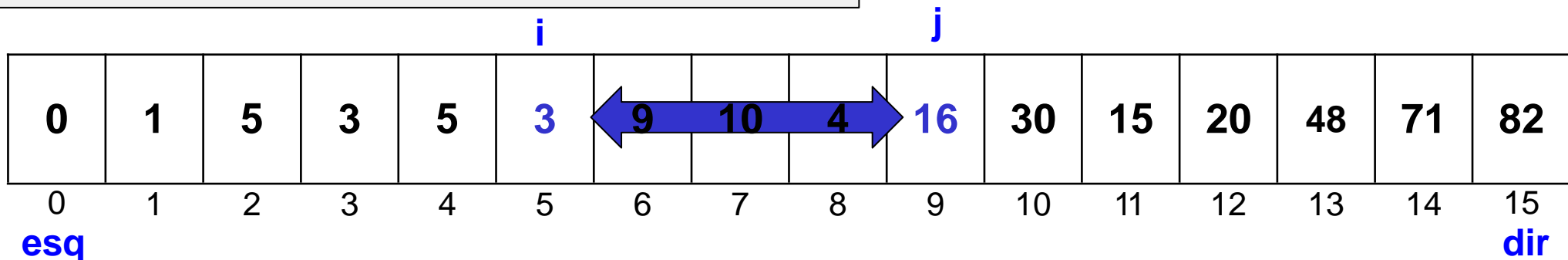
5 <= 9: true

0	1	5	3	5	16	9	10	4	3	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

## Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```



# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

						i	j									
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

# Algoritmo

pivô

10

6 <= 8: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

9 < 10: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							i	j							dir



# Algoritmo

pivô

10

10 < 10: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							i	j							dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

4 > 10: false

							i	j								
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

# Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

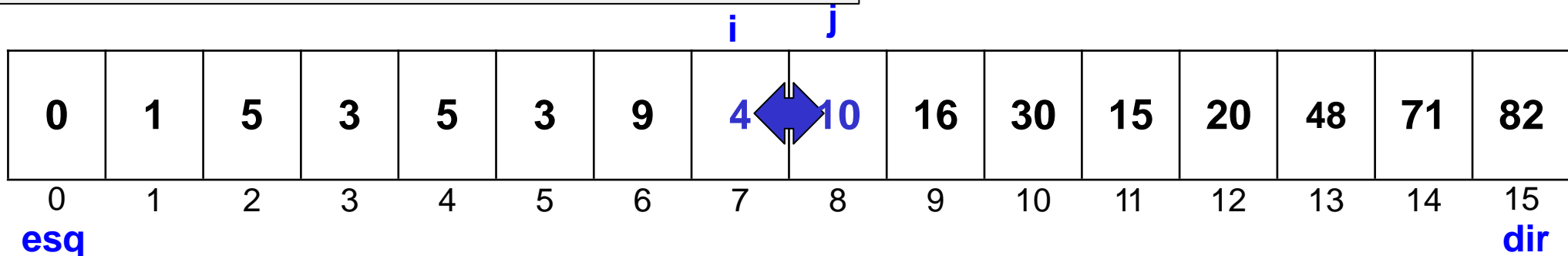
7 <= 8: true

							i	j								
0	1	5	3	5	3	9	10	4	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq																dir

## Algoritmo

pivô 10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```



# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

8 <= 7: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
}
```

pivô

10

if (esq < j)

0 < 7: true

Quicksort(array, esq, j);

if (i < dir)

Quicksort(array, i, dir);

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

pivô

10

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir



# Algoritmo

pivô

?

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

Obs(1): esq e dir são 0 e 7, respectivamente.

Obs(2): o pivô será definido novamente.

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>							<b>dir</b>								

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>							<b>dir</b>								

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
    
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
<b>esq</b>							<b>dir</b>								

# Algoritmo

pivô

3

$$(0 + 7) / 2 = 3$$

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

0 <= 7: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

0 < 3: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

esq

dir

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

1 < 3: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								



# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

5 < 3: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);       i++;       j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

4 > 3: true

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

9 > 3: true

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i

j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

3 > 3: false

i j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq			dir												

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

2 <= 5: true

i j

0	1	5	3	5	3	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

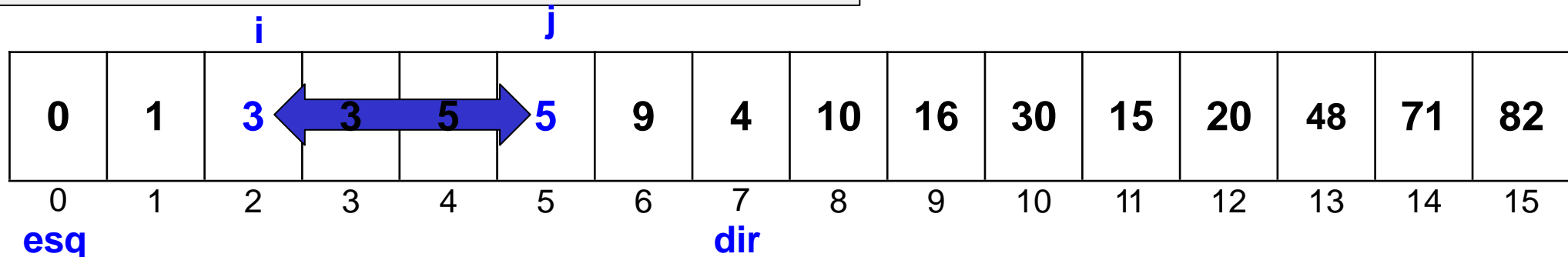


# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```



# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

			i	j												
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
esq							dir									

# Algoritmo

pivô

3

3 <= 4: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

3 < 3: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq			i	j	dir										

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);        i++;    j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

5 > 3: true

			i	j											
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);      i++;      j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);       i++;    j--;  }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

3 > 3: false

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

3 <= 3: true

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								



## Algoritmo

pivô

3

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}

```

i j

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq				dir											

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

Obs: O elemento [3] está na posição correta

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq				dir											

# Algoritmo

pivô

3

4 <= 2: false

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
        {   Trocar(i, j);   i++;   j--;   }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

j

i

0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq				dir											

# Algoritmo

```

void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
    
```

pivô

3

0 < 2: true

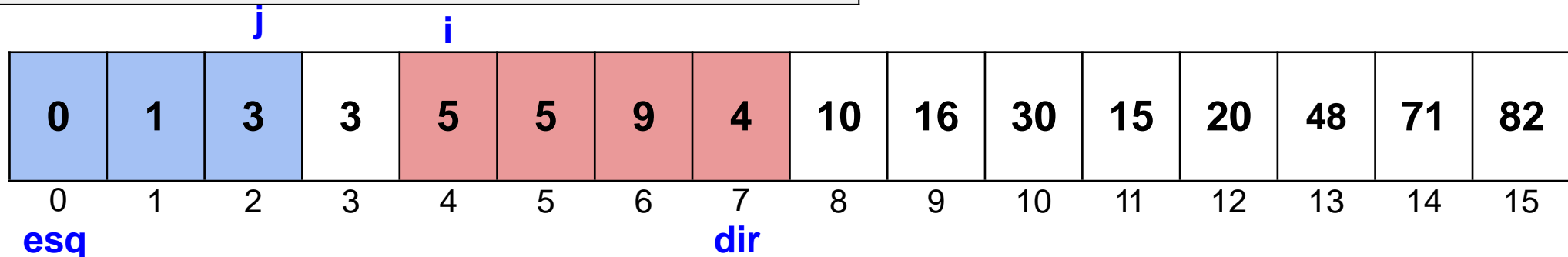
0	1	3	3	5	5	9	4	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq							dir								

# Algoritmo

pivô

3

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```



# Algoritmo

pivô 10

Voltando para a primeira chamada do Quicksort

8 < 15: true

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	3	3	4	5	5	9	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

# Algoritmo

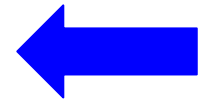
pivô 10

Voltando para a primeira chamada do Quicksort

```
void Quicksort(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[(esq+dir)/2];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j)
            { Trocar(i, j); i++; j--; }
    }
    if (esq < j)
        Quicksort(array, esq, j);
    if (i < dir)
        Quicksort(array, i, dir);
}
```

0	1	3	3	4	5	5	9	10	16	30	15	20	48	71	82
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
esq															dir

- Funcionamento básico
- Algoritmo
- **Análise do número de comparações**
- Escolha do Pivô
- Conclusão



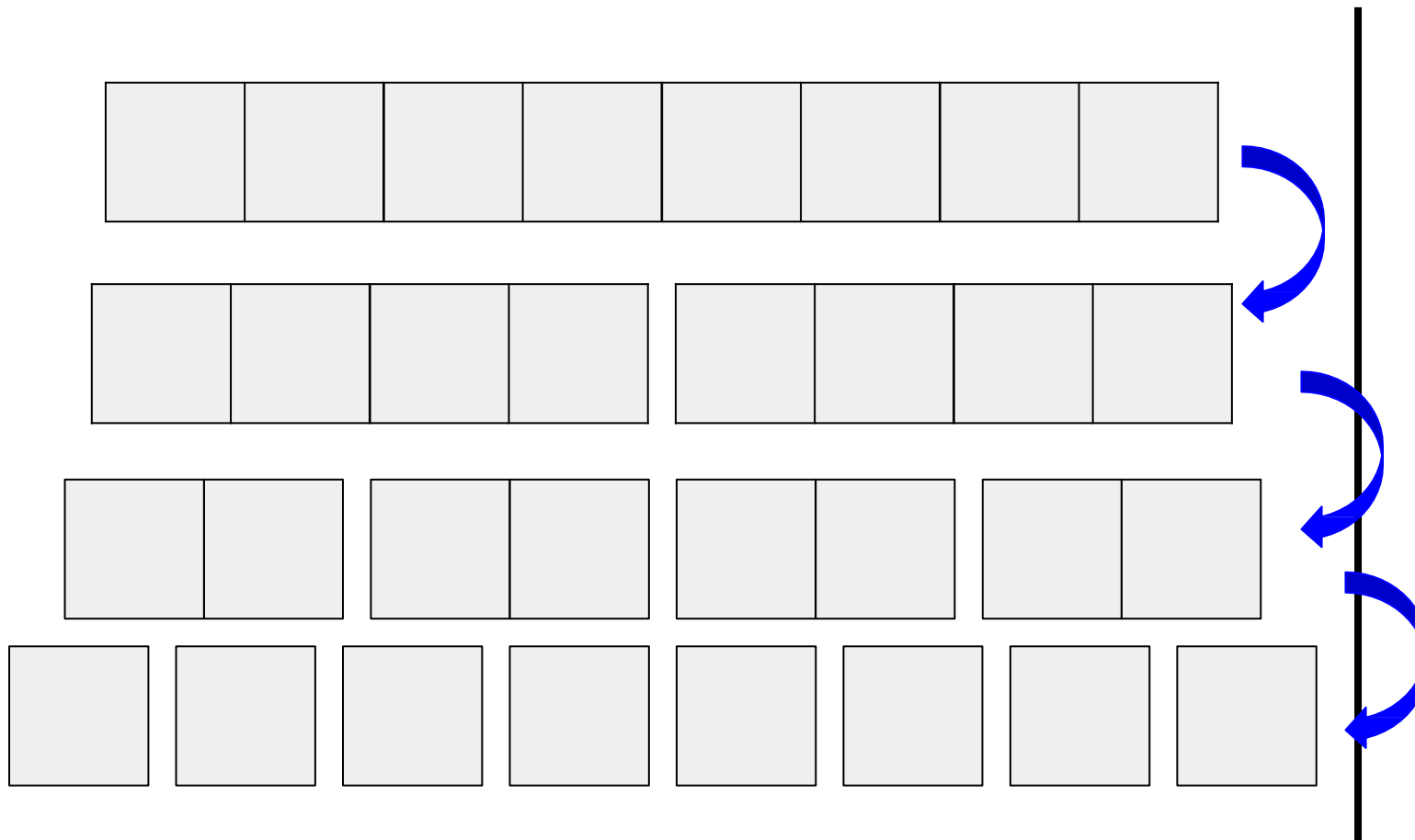


# Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais

# Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais



# Análise do Número de Comparações

- **Melhor caso:** Sistemáticamente, cada partição divide o *array* em duas partes iguais

$$C(n) = n * \lg(n) - n + 1$$

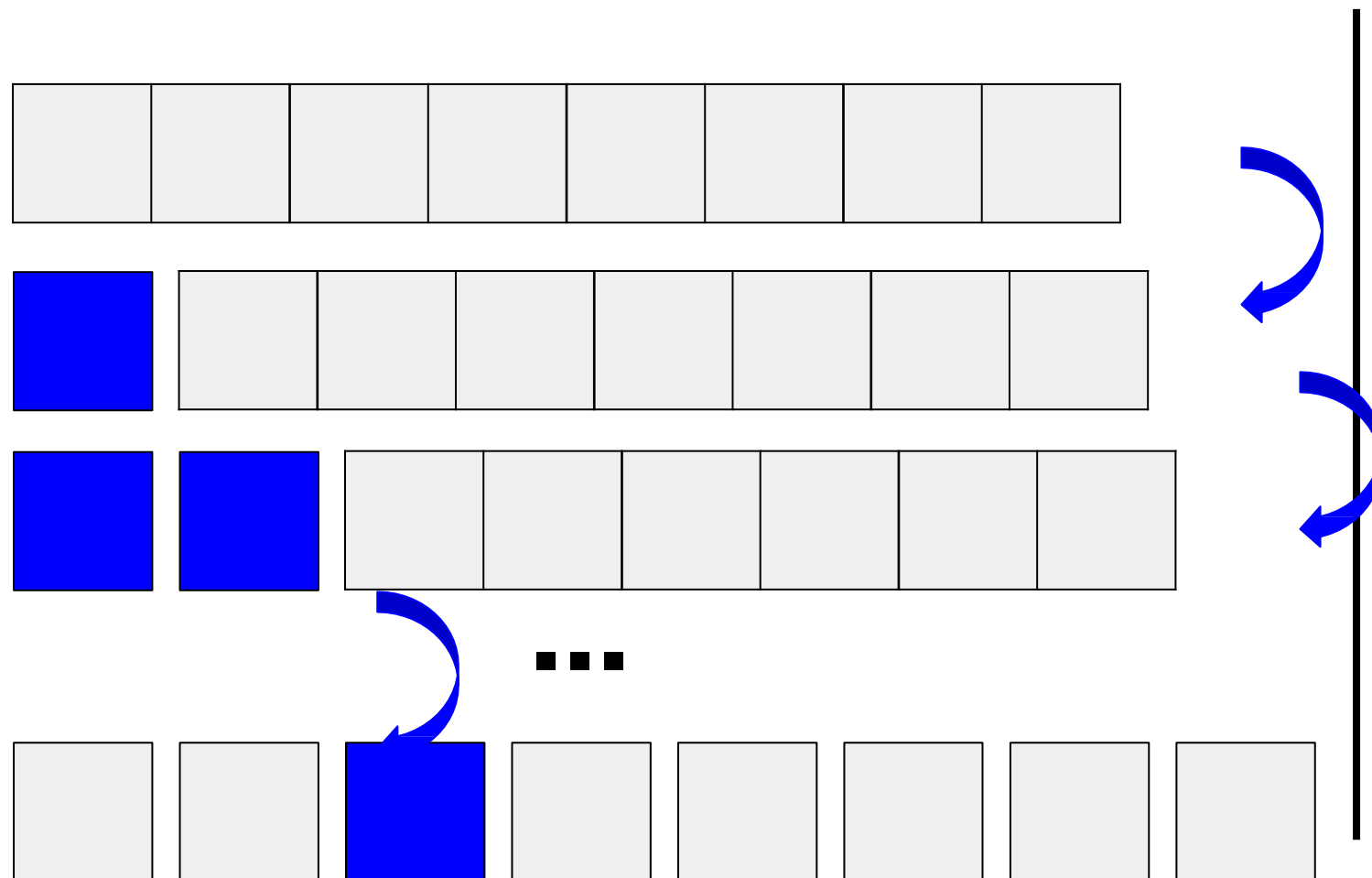
A análise de complexidade do Quicksort depende de equação de recorrência (tópico que será visto mais adiante no curso)

# Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo

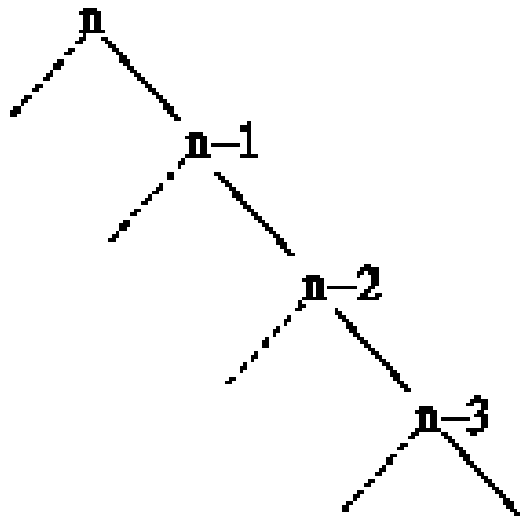
# Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



# Análise do Número de Comparações

- **Pior caso:** Sistemáticamente, o pivô é o menor ou o maior elemento do *array*, eliminando um elemento em cada chamada do algoritmo



$$C(n) = \Theta(n^2)$$

- Existem diversas técnicas para **evitar** o pior caso como, por exemplo, fazer com que o pivô seja a mediana de três elementos do *array*

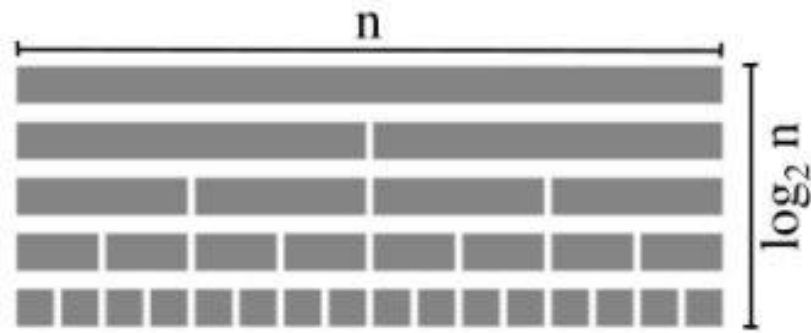
# Análise do Número de Comparações

- **Caso Médio:** Sedgwick e Flajolet (1996, p. 17):

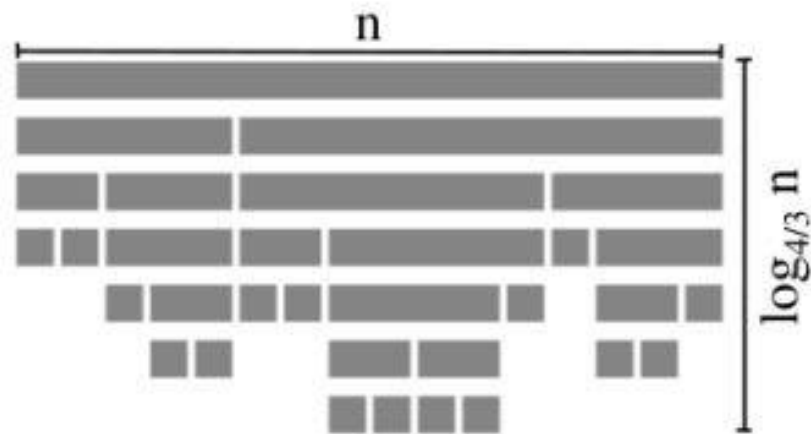
$$C(n) \approx 1,386 * n * \lg(n) - 0,846 * n$$

$O(n \log n)$

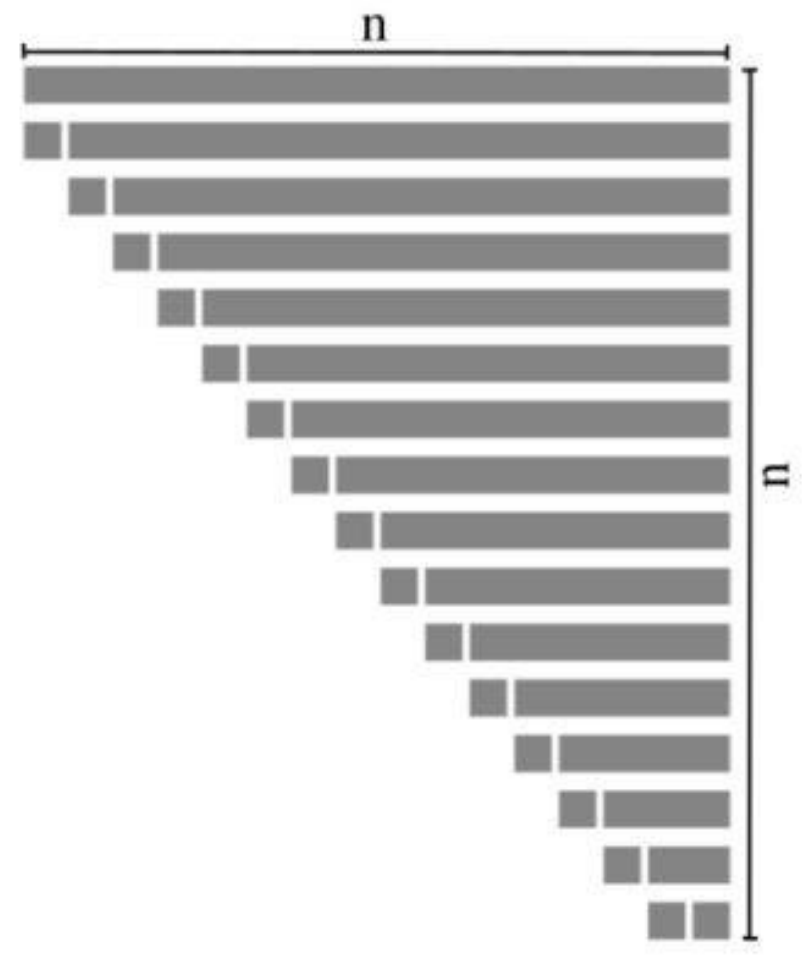
# Análise do Número de Comparações



(a)




(b)



(c)

**Figura:** (a) Melhor caso. (b) Caso médio. (c) Pior caso.



- Funcionamento básico
- Algoritmo
- Análise do número de comparações
- **Escolha do Pivô** 
- Conclusão

# Escolha Aleatória

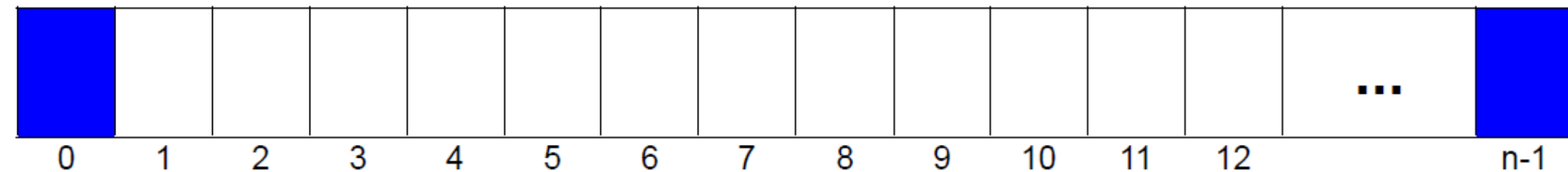
- Escolher aleatoriamente um item da lista como pivô
- Na média, teremos uma partição da lista na proporção:  $\frac{1}{4}$  e  $\frac{3}{4}$
- Se a partição da lista ocorrer pelo menos metade das vezes nessa proporção, o tempo de execução esperado é  $\Theta(n \times \log n)$

# Mediana de Três

- Escolhemos três elementos aleatoriamente
- A mediana dos três será o pivô (isto é, valor do meio)
- Esta estratégia aumenta ainda mais as chances de se obter o caso médio  $\Theta(n \times \log n)$
- Como existe um custo para se obter três elementos aleatórios e obter a mediana, essa estratégia é utilizada apenas em listas “grandes”. Em listas menores, a escolha aleatória simples é mais adequada

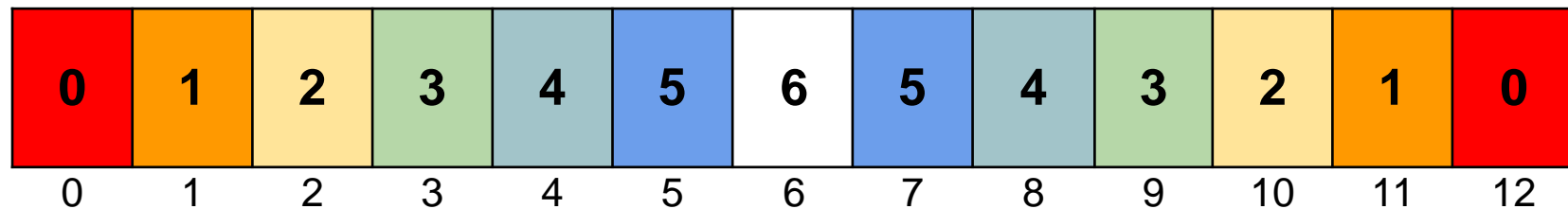
# Primeiro ou Último Elemento


- É uma técnica simples, contudo, se os elementos estiverem em ordem crescente ou decrescente teremos o **pior caso** do Quicksort



# Elemento do Meio

- É uma técnica simples em que o **pior caso** do Quicksort acontece se, sistematicamente, os elementos formarem uma espécie de triângulo



- Funcionamento básico
- Algoritmo
- Análise do número de comparações
- Escolha do Pivô
- **Conclusão** 

# Conclusão

- A razão de sua velocidade é a simplicidade do seu anel interno
- Vantagens:
  - Necessita de apenas uma pequena pilha como memória auxiliar
  - Faz em média  $\Theta(n \times \log(n))$  comparações

- Desvantagens:
  - Seu pior caso para comparações é quadrático
  - Sua implementação é delicada e difícil
  - Método não estável